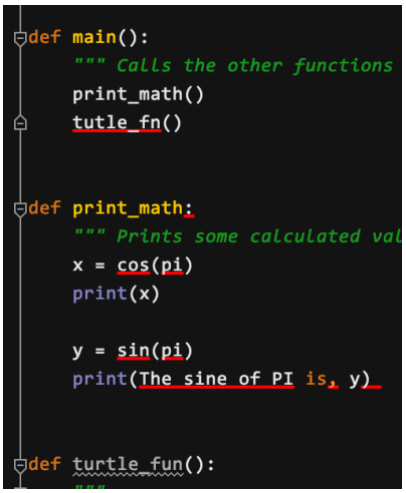
Questions that students frequently wish that they could ask when taking Exam 1

1. I have *red* marks in my code. What do I do?



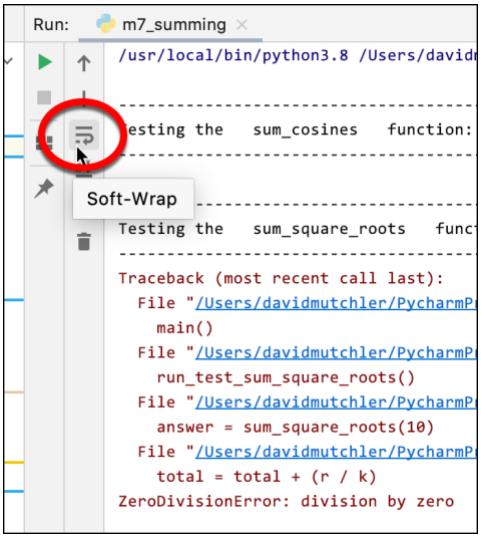
Answer: This indicates a ***syntax*** (notational) ***error*** and must be corrected before attempting to run the module. To do so:

Starting at the top of your code, ***hover over a red mark*** and read the error message; it often (but not always) offers a useful suggestion. Correcting one syntax error may correct some subsequent ones.

Remember that the problem is often on the preceding line (perhaps a missing parenthesis). You can put the cursor beside a parenthesis to see its match highlighted.

Other common sources of error include wrong indentation (being off by even a single space causes errors) and using names that have not been defined in the function in which you are using them.

2. My code produced *red* in the Console when it ran. What do I do?



a. First, ***turn off Soft-Wrap*** and/or resize your Console window to make the error messages easier to read. (See the circled item in the first picture above and to the left.)

b. ***Read the bottommost red error message.*** Sometimes it makes sense, sometimes not. See the next question for some useful error messages.

c. Now, ***turn Soft-Wrap back on*** to make the entirety of the error messages visible.

d. Look at the ***BOTTOM blue line***.

If that line is a line in YOUR module, then ***click on that blue line*** to go straight to the line in your code that caused the program to crash.

Otherwise, if the bottom blue line is in (say) ***rosegraphics.py***, as in the example above and to the right, work your way UP until you reach a blue line that is in YOUR module. Then ***click on the blue line*** that leads to YOUR module. The picture shows skipping over two rosegraphics.py lines and then clicking on the line circled in green.

e. Fix your error, keeping in mind the error message. See the next question for more on fixing the error.

3. My program crashed, but I don't understand what the *red error message* is telling me. Help!

Answer: Some error messages will not be clear until later in the course. But here are some that are commonly encountered on Exam 1 that DO make sense and are very helpful in correcting the error.

a. The message in the example to the right (***'tuple' object has no attribute 'clone'***) often occurs when you send an object of the wrong type when constructing an object, e.g.:

```
  File "/Users/davidmutchler/PycharmProjects/csse120-pub
    super().__init__(center, tkinter.Canvas.create_recta
  File "/Users/davidmutchler/PycharmProjects/csse120-pub
    self.center = center.clone()
AttributeError: 'tuple' object has no attribute 'clone'
```

```
WRONG = rg.Circle((100, 100), 50)
RIGHT = rg.Circle(rg.Point(100, 100), 50)
```

b. Another message that appears frequently is:

```
AttributeError: 'BLAHType' object has no attribute 'XXX'
```

where BLAH and XXX are some particular type and name.

That means that the statement at which the program crashed had a line of the form:

```
... thing.foo ...
```

where ***thing*** is an object of type ***BLAH***. Check that ***thing*** is what you think it is. Check that the attribute in your code (shown as ***XXX*** above) is spelled right. If you don't see the problem, ***print thing*** just before the line that broke to see what it is.

c. Another message that appears frequently is:

```
Type error: unsupported operand type(s) for BLAH: XXX and YYY
```

where *BLAH* is some operator (like +) and *XXX* and *YYY* are some particular types.

```
  /m5_more_graphical_accumulators.py", line 126, in
  draw_squares_from_circle
    new_center = circle.center + 100
TypeError: unsupported operand type(s) for +: 'Point' and 'int'

Process finished with exit code 1
```

That means that the statement at which the program crashed had a line of the form:

... XXX + YYY ... where the things on the left and right side of the operator (here, +) don't make sense to be (in this example) added. In the example above, it does not make sense to add a Point and an integer. Usually you can go to the line at which the program crashed (select the blue link!) and correct the code to what you intended.

4. My program crashed, but I am sure that the line at which the program crashed is correct. What do I do?

The error may or may not be on the line at which your code crashed. Use the *red error message* to guide your thoughts. Use PRINT as needed. (See the next question.)

```
  File "/Users/davidmutchler/PycharmProjects/csse120-public/PythonProjects
  /02-ObjectsFunctionsAndMethods/src/m6_functions_vs_methods.py", line 207
    in try_methods_and_functions
      turtle.forward(100)
TypeError: forward() missing 1 required positional argument: 'distance'
```

One common example of this sort of mistake is shown by the code and error message above and to the right. The error message is not helpful (until Session 9), but the mistake is classic: the author left out the required ***parentheses*** in constructing a SimpleTurtle.

```
window = rg.TurtleWindow()
turtle = rg.SimpleTurtle
turtle.forward(100)
```

5. **My code is failing tests. What do I do?**

First, re-read the ***specification*** that immediately follows the DEF line. ***Make especially sure that you understand the first example in it and WHY the answer in that example is what is given.*** You cannot code a function if you do not understand or misunderstand its specification! If you do not fully understand that example, ask your instructor to walk you through it.

If that does not help, then ***PRINT IS YOUR FRIEND***. Put ***print*** statements where they will help you understand what is going on, and then work through the test that is failing until you see why YOUR code is behaving differently than the example indicates that it should.

```
count = 0
for k in range(n + m - 1):
    if is_prime(n):
        count = count + 1

return count
```

```
Expected: 4
Actual:   11
```

For example, I began with the code above; it ***failed the test*** as shown above and to the right. So, I ***added a print statement*** as shown to the right. It gave the output as show to the far right, from which I deduced that the IF statement was always firing. Finally, I printed the relevant items for the IF statement, as shown in the code below and to the right, and saw my error (the ***n*** should have been ***m + k***).

```
count = 0
for k in range(n + m - 1):
    if is_prime(n):
        count = count + 1
        print("Count is", count)

return count
```

```
Count is 1
Count is 2
Count is 3
Count is 4
Count is 5
Count is 6
Count is 7
Count is 8
Count is 9
Count is 10
Count is 11
Expected: 4
Actual:   11
```

6. **Nothing in showing up in my graphics window. What do I do?**

Did you remember to ***attach*** your objects to the window? To ***render*** the window? Is it possible that your graphics objects have negative coordinate? (Print them to find out.)

```
count = 0
for k in range(n + m - 1):
    print("n, is_prime:", n, is_prime(n))
    if is_prime(n):
        count = count + 1
        print("Count is", count)
```

7. **I called a function but it just did not do anything. What do I do?**

Did you write something like:

```
y = foo
```

when you meant:

```
y = foo()
```

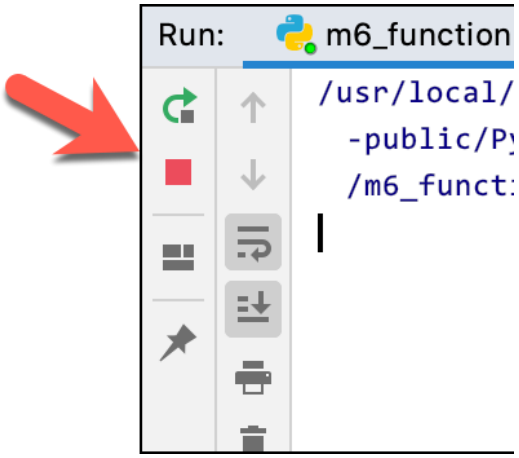
8. **I have a loop that was supposed to make a bunch of objects show up, but only one is appearing. What do I do?**

Might they all be on top of each other? Or off the screen? PRINT the objects to find out.

Also, while it is OK to use *move_by*, it is often used incorrectly. If you used *move_by* and you are seeing only one object show up where you expected many, you may be misusing *move_by*. Replace your *move_by* by equivalent code that constructs objects as needed and see if that helps.

9. **My program is not printing all the tests. What should I do?**

   See if there is a RED SQUARE in the Console (Run) window, as shown to the right. If so, your program is probably in an INFINITE LOOP. Stop the loop by clicking on the red square.



10. **I am doing a summing (or counting) problem. I am sure that I have the general idea right, but I am failing tests. Help!**

    You can use PRINT statements to track down the source of the error, as in item #5 earlier in this document. But before doing that:

    a. ***Re-read the specification, tracing by hand one of the examples***, being sure that you understand and agree with the answer that the function should return on that example.

    b. ***Examine your RANGE statement.*** In the single-argument form, it is always the number of times your loop should run. So:

    - if you want to loop ***z*** times, use ***`range(z)`***.
    - If you want to loop from ***a*** to **b,** including the **b,** use ***`range(b - a + 1)`*** since that is how many numbers there are from ***a to b***. (Try an example if you are unsure about that.)
    - If you want to loop from ***x cubed to y squared***, use ***`range((x ** 3) - (y ** 2))`***.

    c. ***Examine the expression in your loop that is intended to change at each iteration.*** It must at least contain the loop variable (or an auxiliary variable used as a loop variable).

    In the simplest case, where you want to start at ***BLAH*** and go up by 1 each time through the loop, the expression should be **`(BLAH + k)`**. Make sure you understand WHY that is so!